High order iterative methods with memory for nonlinear equations and their dynamics

Cory L. Howk[a], Jose L. Hueso[b], Eulalia Martnez.[b] Carles Teruel[b]

[a] Dept. of Mathematics and Computer Science, W. Carolina University, Cullowhee, North Carolina

[b] Instituto Universitario de Matematica Multidisciplinar. Universitat Politecnica de Valencia, Spain

, ,

Abstract

In this paper we obtain some theoretical results about iterative methods with memory for nonlinear equations. The main idea consists in using for the predictor step of each iteration a quantity that has already been calculated in the previous iteration for the corrector step, usually the quantity governing the slope from the previous corrector step. In this way we do not introduce any extra computation, and more importantly we avoid new functional evaluations, allowing us to obtain high iterative methods in a simple way. A speci c class of methods of this type is introduced, and we prove the convergence order is $2^n + 2^{n-2}$ with $n + 1$ functional evaluations. An exhaustive e ciency study is performed to show the competitiveness of these methods. Finally we test some speci c examples and explore the e ect that this predictor may have on the convergence set by setting a dynamical study.

Key words: Iterative methods with memory; Convergence rate; E ciency; Kung-Traub conjecture; Dynamics.

2000 Mathematics Subject Classi cation: 47H99, 65H10, 65B05, 47H17, 49M15.

This work was supported in part by the project MTM2014-52016-C2-2-P of the Spanish Ministry of Science and Innovation.

1. Introduction

Nonlinear equations appear in a natural way in many applications of science and engineering. The solutions are typically approximated via iterative methods. Newton's

Email addresses: clhowk@email.wcu.edu , jlhueso@mat.upv.es , eumarti@mat.upv.es (Carles Teruel).

method and other \simple" techniques are common, and historically have been sufficient. However, nowadays high-order methods are very important, as many scientific applications need high precision in their computations. For example, in [1], high precision calculations are used to solve interpolation problems in astronomy, whereas in [2] the authors describe the use of arbitrary precision computations to improve the results obtained in climate simulations. The precision required would necessitate a large number of iterations if one utilized a low-order method. The results of these numerical experiments in [1,2] show that the high-order methods associated with multiprecision floating point arithmetic are very useful, as they yield a clear reduction in iterations.

The convergence order of an iterative method is directly related with the efficiency in the sense of the conjecture of Kung-Traub, [7], which established the optimal relation between these concepts for the case of iterative methods without memory. In short, the conjecture states that a method without memory that uses $n + 1$ function evaluations per iterate can have a convergence rate of at most 2^n. It is important to note that this conjecture does not hold for methods that incorporate memory.

This paper focuses on the consequences of memory reuse rather than just using the historical points in new ways. It is well known that iterative methods with memory can surpass the Kung-Traub conjecture, but often the computational cost of obtaining high-order methods is very expensive. Therefore we concentrate in this work on defining predictor-corrector iterative methods with memory in a simple way; that is, trying to use memory to obtain a high-order of convergence, but without introducing extra computation to the iterative expression.

Different studies about iterative methods with memory have recently been published. See among others, [9]-[16]. Here we deal with general schemes of k steps which allow us to obtain different convergence orders depending on the number of steps performed. In this sense we recall paper [17], where an iterative method without memory is constructed, which utilizes $n + 1$ function evaluations per iterate and has optimal convergence order. Different improvements have been published based on this scheme. One of these in particular, [15], is a complete study where the authors construct a biparametric family of iterative methods with memory where, by using memory to approximate one of the parameters, the convergence order improves to $2^n + 2^{n-1}$ and by approximating the second parameter they reach $2^n + 2^{n-1} + 2^{n-2}$. In both cases the number of functional evaluations is maintained at $n + 1$.

Nevertheless, the computations made in order to reach these high convergence orders has been costly. Our idea in this work is to simplify this cost by constructing high-order methods in a way that the iterative expression remains as simple as possible. We use the idea of using for the predictor step of each iteration a quantity that has already been calculated in the previous iteration for the corrector step, usually the quantity governing the slope from the previous corrector step. In this way we do not introduce extra computation.

The predictor-corrector iterative methods constructed are an alternative that can be competitive, because while high-order methods are important, the operational cost of getting them must also be taken into account. For this reason an exhaustive efficiency study is performed to show the effectiveness of these methods.

In the next section we introduce the classes of the predictor-corrector algorithms with memory that we will be studying. In section 3, convergence order results are obtained and some initial results about efficiency are derived. Section 4 is devoted to introducing the new multistep iterative method with memory and the comparison with recently published iterative methods with similar characteristics, including a subsection of numerical results. Finally, in section 5, we explore the e ect that this kind of construction for deriving methods may have on the convergence set by setting a dynamical study.

2. Predictor-Corrector iterative methods

While Newton's Method is one of the most popular root- nding algorithms due both to its simplicity and its quadratic rate of convergence, many more powerful methods exist with faster convergence rates. The methods of derivation for these powerful methods often result in implicit equations. For example, if one integrates $f'(x)$ between the current iterate and the (unknown) root, $\int_{x_n} f'(x)dx = \ f(x_n)$ and apply the Midpoint Rule to the integral, we get $f'(\frac{1}{2}[x_n + \ ])(\ x_n) \ f(x_n)$, which can be rearranged into the implicit equation $\ x_n \ \frac{f(x_n)}{f'(\frac{1}{2}[x_n + \ ])}$. These equations are typically applied algorithmically as a predictor-corrector set ([4]-[6]); for example,

$$y_n = x_n \quad f(x_n)(\ x_n)$$
$$x_{n+1} = x_n \quad \frac{f(x_n)}{f'(\frac{1}{2}[x_n + y_n])} \tag{1}$$

for some function . It is common to use Newton's Method for the predictor step,

$$y_n = x_n \quad \frac{f(x_n)}{f'(x_n)}$$
$$x_{n+1} = x_n \quad f(x_n)(\ x_n; y_n): \tag{2}$$

In this sense, one could view this type of algorithm as Newton's Method with a corrective factor. However, if $(\ x_n; y_n)$ does not incorporate $f'(x_n)$, then this predictor has increased the number of functional evaluations required per iterate. Rather than introduce this extra computation, we would like to use for the predictor step a quantity that has already been calculated, namely $(x_{n\ 1}; y_{n\ 1})$, that is the quantity governing the slope from the previous corrector step.

$$y_n = x_n \quad f(x_n)(\ x_{n\ 1}; y_{n\ 1})$$
$$x_{n+1} = x_n \quad f(x_n)(\ x_n; y_n): \tag{3}$$

This type of method will be referred to as the Standard 2-step Predictor-Corrector algorithm, denoted by SA:

However, since y_n is already likely an improvement on x_n, one could also incorporate $f(y_n)$ into the algorithm. This is commonly performed through Newton's Method, and we denote this method by NP:

$$y_n = x_n \quad \frac{f(x_n)}{f'(x_n)}$$
$$x_{n+1} = y_n \quad f(y_n)(\ x_n; y_n): \tag{4}$$

We again consider the effect of incorporating the previous slope into the predictor step

$$\begin{aligned} y_n &= x_n - f(x_n)\,\phi(x_{n-1}; y_{n-1}) \\ x_{n+1} &= y_n - f(y_n)\,\phi(x_n; y_n): \end{aligned} \tag{5}$$

and refer to this as the Improved 2-step Predictor-Corrector algorithm, denoted by IA. This can be further expanded into an Improved multi-step Predictor-Corrector algorithm, denoted by IMS and given by:

$$\begin{aligned} y_n^{(0)} &= x_n - f(x_n)\,\phi(x_{n-1}; y_{n-1}^{(0)}; y_{n-1}^{(1)}; \ldots; y_{n-1}^{(k)}) \\ y_n^{(i)} &= y_n^{(i-1)} - f(y_n^{(i-1)})\,\phi_i(x_n; y_n^{(0)}; y_n^{(1)}; \ldots; y_n^{(i-1)}); \quad i = 1; 2; \ldots; k \\ x_{n+1} &= y_n^{(k)} - f(y_n^{(k)})\,\phi(x_n; y_n^{(0)}; y_n^{(1)}; \ldots; y_n^{(k)}); \end{aligned} \tag{6}$$

Notice that in the second and successive iterations, the first step is using the slope from the last step giving a method with memory; that is both are using function ϕ, while intermediate steps use different functions ϕ_i:

3. Convergence Analysis

We begin by establishing convergence results for the two-step predictor-corrector iterative schemes previously defined to locate a zero $x = \alpha$ of $f(x)$. Define the error terms $\varepsilon_n = x_n - \alpha$ and $\tilde{\varepsilon}_n = y_n - \alpha$ for all n. The sequence $\{x_n\}_{n=0}^{\infty}$ is said to converge to $x = \alpha$ with rate of convergence β if $\lim_{n\to\infty} \frac{|\varepsilon_{n+1}|}{|\varepsilon_n|^{\beta}} = \gamma$ for positive constants γ, β. The relationship between ε_{n+1} and ε_n can therefore be written $\varepsilon_{n+1} = A_n \varepsilon_n^{\beta} + h.o.t.$ where $A_n \to \gamma$ as $n \to \infty$ and h.o.t. denotes higher-order terms which satisfy $\lim_{n\to\infty} \frac{h.o.t.}{\varepsilon_n^{\beta}} = 0$. This relationship will be denoted by $\varepsilon_{n+1} \sim \varepsilon_n^{\beta}$. The specific terms in h.o.t. will not be important in the subsequent analysis.

Theorem 1. Suppose that the error condition for the corrector step of the Standard 2-step Predictor-Corrector algorithm is given by $\varepsilon_{n+1} \sim \varepsilon_n^p(\tilde{\varepsilon}_n)^q$, with p and q real positive numbers, and that f is sufficiently differentiable. Then the convergence rate for the algorithm is given by $\beta = \dfrac{p + q + \sqrt{(p+q)^2 + 4q}}{2}$.

Proof. By hypothesis we have that equation (3) satisfies the condition $\varepsilon_{n+1} \sim \varepsilon_n^p(\tilde{\varepsilon}_n)^q$, so that $\varepsilon_{n+1} = A_n \varepsilon_n^p(\tilde{\varepsilon}_n)^q + h.o.t.$ Assuming f is sufficiently differentiable, then after subtracting α from both sides of equation (3), one has:

$$\begin{aligned} \varepsilon_{n+1} &= \varepsilon_n - f(x_n)\,\phi(x_n; y_n) \qquad (7)\\ &= \varepsilon_n - \Big[\sum_{i=0}^{\ell-1} \frac{f^{(i)}(\alpha)}{i!}\varepsilon_n^i + \frac{f^{(\ell)}(\xi_1)}{\ell!}\varepsilon_n^{\ell}\Big]\phi(x_n; y_n) \\ &= \varepsilon_n \Big[1 - \Big(\sum_{i=1}^{\ell-1} \frac{f^{(i)}(\alpha)}{i!}\varepsilon_n^{i-1} + \frac{f^{(\ell)}(\xi_1)}{\ell!}\varepsilon_n^{\ell-1}\Big)\phi(x_n; y_n)\Big] \end{aligned}$$

for some $\xi_1 \in (x_n; \alpha)$, and therefore,

$$\alpha(x_n;y_n) = \frac{1 - A_n\varepsilon_n^{p-1}(\varepsilon_n)^q + h.o.t.}{\sum_{i=1}^{\ell-1}\frac{f^{(i)}(\alpha)}{i!}\varepsilon_n^{i-1} + \frac{f^{(\ell)}(\xi_1)}{\ell!}\varepsilon_n^{\ell-1}}.$$

Considering the next predictor step, after subtracting α from both sides there exists $\xi_2 \in (x_{n+1};\alpha)$ such that

$$\begin{aligned}
\varepsilon_{n+1} &= \varepsilon_{n+1} - f(x_{n+1})\,\alpha(x_n;y_n)\\
&= \varepsilon_{n+1} - \varepsilon_{n+1}\left[\sum_{i=1}^{\ell-1}\frac{f^{(i)}(\alpha)}{i!}\varepsilon_{n+1}^{i-1} + \frac{f^{(\ell)}(\xi_2)}{\ell!}\varepsilon_{n+1}^{\ell-1}\right]\frac{1 - A_n\varepsilon_n^{p-1}(\varepsilon_n)^q + h.o.t.}{\sum_{i=1}^{\ell-1}\frac{f^{(i)}(\alpha)}{i!}\varepsilon_n^{i-1} + \frac{f^{(\ell)}(\xi_1)}{\ell!}\varepsilon_n^{\ell-1}}\\
&= \varepsilon_{n+1}\left[1 - \left(1 + \frac{f''(\alpha)}{2f'(\alpha)}(\varepsilon_{n+1} - \varepsilon_n) + h.o.t.\right)(1 - A_n\varepsilon_n^{p-1}(\varepsilon_n)^q + h.o.t.)\right]\\
&= \varepsilon_{n+1}\left[A_n\varepsilon_n^{p-1}(\varepsilon_n)^q - \frac{f''(\alpha)}{2f'(\alpha)}(\varepsilon_{n+1} - \varepsilon_n) + A_n\frac{f''(\alpha)}{2f'(\alpha)}\varepsilon_n^{p-1}(\varepsilon_n)^q(\varepsilon_{n+1} - \varepsilon_n) + h.o.t.\right]\\
&= B_n\varepsilon_{n+1}\varepsilon_n + h.o.t.
\end{aligned}$$

Therefore, $\varepsilon_n \sim \varepsilon_n\varepsilon_{n-1}$ and $\varepsilon_{n+1} \sim \varepsilon_n^p(\varepsilon_n)^q = \varepsilon_n^{p+q}\varepsilon_{n-1}^q$. Defining $\delta_n = \ln(\varepsilon_n)$ for all n, the convergence rate is given by the characteristic equation:

$$\delta_{n+1} - (p+q)\delta_n - q\delta_{n-1} = 0$$

which has roots $\frac{p+q \pm \sqrt{(p+q)^2+4q}}{2}$, and therefore the rate of convergence is given by $\rho = \frac{p+q+\sqrt{(p+q)^2+4q}}{2}$. □

3.1. Applying theorem 1 to the Midpoint method with memory

We consider the Midpoint Method with memory, denoted by MP, derived at the beginning of Section 2. It is given by:

$$\begin{aligned}
y_n &= x_n - \frac{f(x_n)}{f'(\frac{1}{2}[x_{n-1} + y_{n-1}])}\\
x_{n+1} &= x_n - \frac{f(x_n)}{f'(\frac{1}{2}[x_n + y_n])}.
\end{aligned} \tag{8}$$

This is a Standard 2-step Predictor-Corrector algorithm. The error relationship from the corrector step is derived in [10] and shown to be $\varepsilon_{n+1} \sim \varepsilon_n\varepsilon_n$, so that $p = q = 1$. Therefore, according to Theorem 1, the rate of convergence is $\rho = \frac{1+1+\sqrt{(1+1)^2+4}}{2} = 1+\sqrt{2} \approx 2.414$. We note that since this algorithm utilizes two new functional evaluations per iterate, the optimal method without memory could only achieve a convergence rate of 2 according to the Kung-Traub conjecture.

Theorem 2. Suppose that the error condition for the corrector step of the Improved 2-step Predictor-Corrector algorithm is given by $\varepsilon_{n+1} \sim \varepsilon_n^p(\varepsilon_n)^q$, with p and q real positive numbers, and f is sufficiently differentiable. Then the convergence rate for the algorithm, denoted by IA, is given by $\rho = \frac{(p+q+1)+\sqrt{(p+q+1)^2 - 4p}}{2}$.

Proof. The proof is the same as Theorem 1, except that equation (7) will be:

$$\varepsilon_{n+1} = \varepsilon_n - f(y_n)\,\varphi(x_n; y_n);$$

resulting in

$$\varphi(x_n; y_n) = \frac{1 - A_n \varepsilon_n^p (\delta_n)^{q-1} + h.o.t.}{\sum_{i=1}^{\ell-1} \frac{f^{(i)}(\alpha)}{i!}(\delta_n)^{i-1} + \frac{f^{(\ell)}(\xi_1)}{\ell!}(\delta_n)^{\ell-1}};$$

and consequently, $\delta_n \approx \varepsilon_n \delta_{n-1}$. To determine the convergence rate, we must therefore consider the system:

$$\begin{aligned} \delta_n &= \varepsilon_n \delta_{n-1} \\ \varepsilon_{n+1} &= \varepsilon_n^p (\delta_n)^q. \end{aligned}$$

Setting $\eta_n = \ln(\delta_n)$ and $\xi_n = \ln(\varepsilon_n)$ for all n,

$$\begin{aligned} \eta_n &= \xi_n + \eta_{n-1} \\ \xi_{n+1} &= p\,\xi_n + q\,\eta_n. \end{aligned}$$

By noting that $\eta_n = \frac{\xi_{n+1} - p\,\xi_n}{q}$, the characteristic equation is found to be:

$$\xi_{n+1} - (p + q + 1)\,\xi_n + p\,\xi_{n-1} = 0;$$

which has solutions $\lambda = \frac{(p + q + 1) \pm \sqrt{(p + q + 1)^2 - 4p}}{2}$, and therefore the rate of convergence is given by $\rho = \frac{(p + q + 1) + \sqrt{(p + q + 1)^2 - 4p}}{2}$.

□

3.2. Applying theorem 1 and theorem 2 for different values of p and q

The convergence rates of algorithms of the standard and improved types have been obtained for various combinations of p and q and can be seen in Table 1, along with the corresponding convergence rates for algorithms utilizing a Newton predictor as shown in Systems (2) and (4). Notice that for the methods for which we use Newton's method as a predictor we assume the error condition for the corrector step $\varepsilon_{n+1} \approx \varepsilon_n^p (\delta_n)^q$, resulting in $\varepsilon_{n+1} \approx \varepsilon_n^p (\varepsilon_n^2)^q$ so that the order is $2q + p$. We can see in the results of Table 1 that this method is always reaching the higher convergence rate. However, the convergence rate must be related to the efficiency index as is done in the next section.

3.3. Efficiency Indices

The efficiency index of an iterative algorithm as defined in [6] measures the balance between the rate of convergence (ρ) and the number of functional evaluations (m) required per iterate, and is defined by $EI = \rho^{1/m}$. Obviously methods are preferable when they have a higher efficiency index.

The algorithms stated in the previous section, SA and IA, have slower convergence order than their Newton analogs, NP. However, one must also consider the computational

Method	qnp	1	2	3	4	5
N		3	4	5	6	7
SA	1	2.414	3.303	4.236	5.193	6.162
IA		2.618	3.414	4.303	5.236	6.193
NP		5	6	7	8	9
SA	2	3.562	4.449	5.372	6.317	7.275
IA		3.732	4.562	5.449	6.372	7.317
NP		7	8	9	10	11
SA	3	4.646	5.531	6.464	7.405	8.359
IA		4.791	5.646	6.541	7.464	8.405
NP		9	10	11	12	13
SA	4	5.702	6.606	7.531	8.472	9.424
IA		5.828	6.702	7.606	8.531	9.472
NP		11	12	13	14	15
SA	5	6.742	7.653	8.583	9.525	10.477
IA		6.854	7.742	8.653	9.583	10.525

Table 1
Convergence rates for values of p and q.

work required for calculating each iterate. While replacing the slope from the initial predictor step with the slope from the previous corrector step often results in a slight decrease in convergence, it may result in one fewer functional evaluation. This is very important since it can therefore allow algorithms to surpass the optimal convergence rate for nonmemory algorithms as stated by the Kung-Traub conjecture. The following theorem addresses the e ciency of the respective algorithms.

Theorem 3. Suppose one iterative scheme has convergence rate $_1$, requiring m functional evaluations per iterate, while a second iterative scheme has convergence rate $_2$ with $1 < \ _2 < \ _1$, requiring $m \ \ 1$ functional evaluations per iterate. The e ciency index of the second scheme is greater than the rst scheme for $m < \frac{\ln(\ _1)}{\ln(\ _1) \ \ \ln(\ _2)}$.

Proof. We would like to know the values of m for which $\ _1^{1=m} < \ _2^{1=(m \ \ 1)}$. Taking the natural log of both sides,

$$\frac{1}{m}\ln(\ _1) < \frac{1}{m \ \ 1}\ln(\ _2)$$
$$\frac{\ln(\ _1)}{\ln(\ _2)} < \frac{m}{m \ \ 1} = 1 + \frac{1}{m \ \ 1}$$
$$m \ \ 1 < \frac{1}{\frac{\ln(\ _1)}{\ln(\ _2)} \ \ 1} = \frac{\ln(\ _2)}{\ln(\ _1) \ \ \ln(\ _2)}$$

which, if we add 1 to both sides, is equivalent to

$$m < \frac{\ln(\ _1)}{\ln(\ _1) \ \ \ln(\ _2)}:$$

□

According to this Theorem, the Standard 2-step Predictor-Corrector is more efficient than its Newton analog when:

$$m < \frac{\ln(p+2q)}{\ln\left(\frac{2(p+2q)}{p+q+\sqrt{(p+q)^2+4q}}\right)};$$

while the Improved 2-step Predictor-Corrector is more efficient than its Newton analog when:

$$m < \frac{\ln(p+2q)}{\ln\left(\frac{2(p+2q)}{p+q+1+\sqrt{(p+q+1)^2-4p}}\right)}:$$

In Figure 1 we observe the number of functional evaluations (m) for which standard PC (SA) is more efficient than its Newton analog, provided the "slower" algorithm utilizes one fewer function evaluation per iterate than the "faster" algorithm. The corresponding values for the Improved PC (IA) can be observed in Figure 2. In each case, the real value for the improved PC was greater than the standard PC, although the floor value was the same in many cases.

Figure 1. Values of m for which $EI(SA) > EI(NP)$.

Figure 2. Values of m for which $EI(IA) > EI(NP)$.

Next we consider the convergence rate of the Improved multi-step Predictor-Corrector (IMS) defined by (6).

Theorem 4. Suppose that the error conditions for the predictor steps $y^{(1)}, \ldots, y^{(k)}$ of the Improved k-step Predictor-Corrector algorithm are given by $\epsilon_n^{(\ell)} \approx \epsilon_n^{r_\ell}(\epsilon_n^{(0)})^{s_\ell}$, that of the corrector step is given by $\epsilon_{n+1} \approx \epsilon_n^{p}(\epsilon_n^{(0)})^{q}$, and that f is sufficiently differentiable. Then the convergence rate for the algorithm is given by:

$$\rho = \frac{(p+q+s_k) + \sqrt{(p+q+s_k)^2 - 4(ps_k - qr_k)}}{2}:$$

Proof. This proof begins the same as Theorems 1 and 2, except that equation (7) will be $\epsilon_{n+1} = \epsilon_n^{(k)} - f(y_n^{(k)})/\phi(x_n; y_n^{(0)}; y_n^{(1)}; \ldots; y_n^{(k)})$, resulting in